Resource-Aware Scheduling

Kalyan Muthukumar
Intel Compiler Labs, Bangalore
(Joint work with D. Lavery, G. Hoflehner, C. Lim, and J.F. Collard)

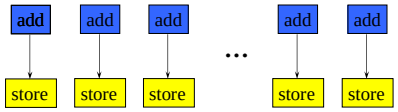
Current schedulers

- Mostly focus on dependence height
- Some use ad-hoc heuristics for taking into account resources (for sched priority)
- No algorithmic approach to take into account resources (for sched priority)
- Suboptimal schedules for resource-bound scheduling regions
- Problem even for EPIC architectures

Our contribution

- Algorithm that takes into account resources while scheduling (for sched priority)
- Implemented in the Intel® product compiler for Itanium®
- Results

Example Scheduling Region



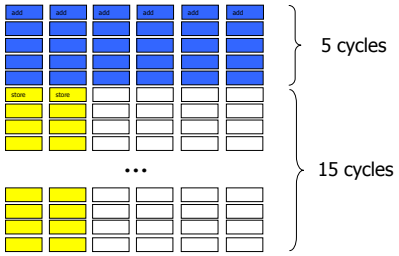
- 30 adds feeding 30 stores
- Resource-bound, not dependence-height bound
- Dependence Height of adds = 1
- Dependence Height of stores = 0

Intel® Itanium® 2 μarch

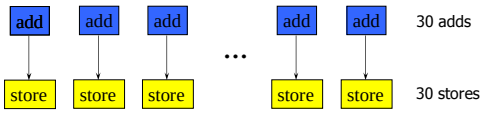
- Issue width = 6 instructions (2 bundles)
- 4 memory ops per cycle (2 loads + 2 stores)
- 2 Integer (ALU) ops per cycle
- 2 Floating Point ops per cycle
- 3 Branch ops per cycle

- Most Memory/Integer ops have latency = 1

Schedule w/o Resource-Awareness



Definitions



For a scheduling region B,

DepLength(B) = Total Height of Dependence Graph + 1 = 1+1 = 2 cycles

ResLength(B) = Maximum number of cycles that any one resource needs = 15 cycles

MinSchedLength(B) = Max (DepLength(B), ResLength(B)) = Max(2, 15) = 15 cycles

ActualSchedLength(B) = Actual number of cycles for block after scheduling
This is >= MinSchedLength(B)

Definitions…

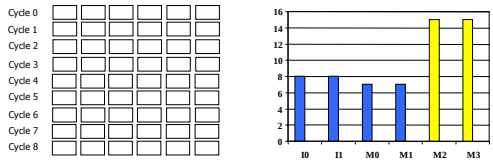
DepDeadline(I) = MinSchedLength(B) - DepHeight(I) – 1
ResDeadline(I) = MinSchedLength(B) - ResHeight(InstrClass(I))
Deadline(I) = min (DepDeadline(I), ResDeadline(I))

Slack(I) = Deadline(I) - current_cycle

- From ready list, pick instruction with least slack

Cycle 0 (before scheduling)

Ready List → 30 adds

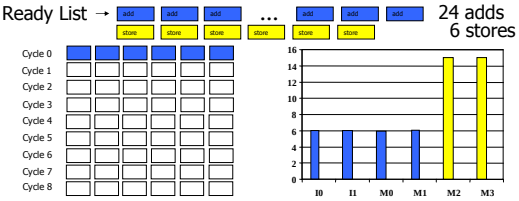


DepHeight(A) = 1, ResHeight(A) = 8, DepDeadline(A) = 15-1-1 = 13

ResDeadline(A) = 15-8 = 7, Deadline(A) = min(7, 13) = 7, **Slack(A) = 7**

Cycle 1 (before scheduling)

Ready List → 24 adds 6 stores



DepHeight(A) = 1, ResHeight(A) = 6, DepDeadline(A) = 15-1-1 = 13

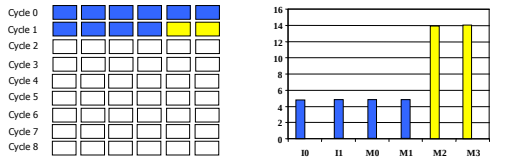
ResDeadline(A) = 15-6 = 9, Deadline(A) = min(9, 13) = 9, **Slack(A) = 8**

DepHeight(S) = 0, ResHeight(S) = 15, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-15 = 0, Deadline(S) = min(0, 14) = 0, **Slack(S) = -1**

Cycle 2 (before scheduling)

Ready List → 20 adds 8 stores



DepHeight(A) = 1, ResHeight(A) = 5, DepDeadline(A) = 15-1-1 = 13

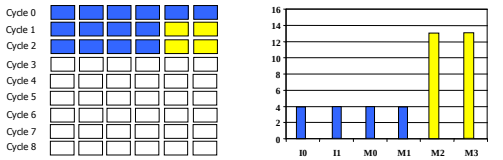
ResDeadline(A) = 15-5 = 10, Deadline(A) = min(10, 13) = 10, **Slack(A) = 8**

DepHeight(S) = 0, ResHeight(S) = 14, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-14 = 1, Deadline(S) = min(1, 14) = 1, **Slack(S) = -1**

Cycle 3 (before scheduling)

Ready List → 16 adds 10 stores



DepHeight(A) = 1, ResHeight(A) = 4, DepDeadline(A) = 15-1-1 = 13

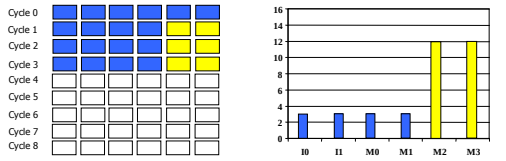
ResDeadline(A) = 15-4 = 11, Deadline(A) = min(11, 13) = 10, **Slack(A) = 8**

DepHeight(S) = 0, ResHeight(S) = 13, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-13 = 2, Deadline(S) = min(2, 14) = 2, **Slack(S) = -1**

Cycle 4 (before scheduling)

Ready List → 12 adds 12 stores



DepHeight(A) = 1, ResHeight(A) = 3, DepDeadline(A) = 15-1-1 = 13

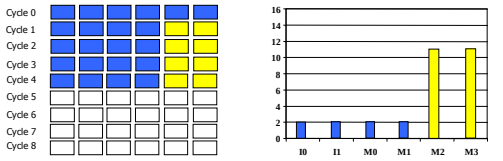
ResDeadline(A) = 15-3 = 12, Deadline(A) = min(12, 13) = 12, **Slack(A) = 8**

DepHeight(S) = 0, ResHeight(S) = 12, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-12 = 3, Deadline(S) = min(3, 14) = 3, **Slack(S) = -1**

Cycle 5 (before scheduling)

Ready List → 8 adds 14 stores



DepHeight(A) = 1, ResHeight(A) = 2, DepDeadline(A) = 15-1-1 = 13

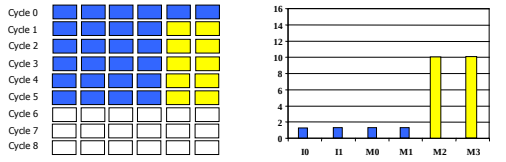
ResDeadline(A) = 15-2 = 13, Deadline(A) = min(13, 13) = 13, **Slack(A) = 8**

DepHeight(S) = 0, ResHeight(S) = 11, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-11 = 4, Deadline(S) = min(4, 14) = 4, **Slack(S) = -1**

Cycle 6 (before scheduling)

Ready List → 4 adds 16 stores



DepHeight(A) = 1, ResHeight(A) = 1, DepDeadline(A) = 15-1-1 = 13

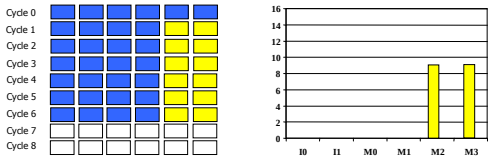
ResDeadline(A) = 15-1 = 14, Deadline(A) = min(14, 13) = 13, **Slack(A) = 7**

DepHeight(S) = 0, ResHeight(S) = 10, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-10 = 5, Deadline(S) = min(5, 14) = 5, **Slack(S) = -1**

Cycle 7 (before scheduling)

Ready List → 18 stores

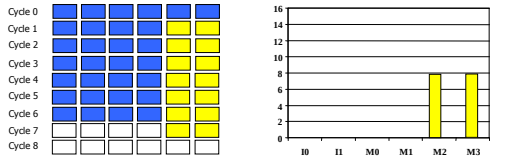


DepHeight(S) = 0, ResHeight(S) = 9, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-9 = 6, Deadline(S) = min(6, 14) = 6, **Slack(S) = -1**

Cycle 8 (before scheduling)

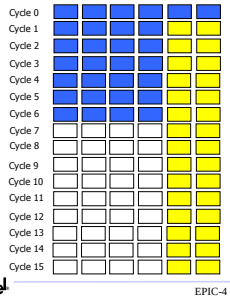
Ready List → 16 stores



DepHeight(S) = 0, ResHeight(S) = 8, DepDeadline(S) = 15-1-0 = 14

ResDeadline(S) = 15-8 = 7, Deadline(S) = min(7, 14) = 7, **Slack(S) = -1**

Final schedule



Schedule with resource-awareness = 16 cycles

Schedule w/o resource-awareness = 20 cycles

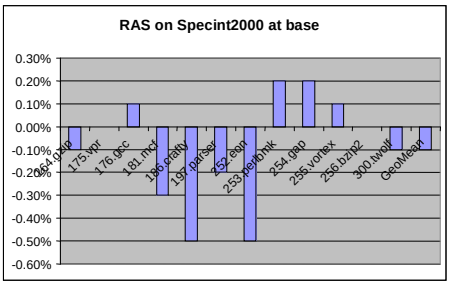
Gain = 20%

RAS Algorithm

```
Initialize the Ready Instructions in the DAG;
Compute their Slack and sort them according to their Slack;

While (Ready List is not Empty) {
       Remove an Instruction from Ready List that has the
       least slack and that can be scheduled in current cycle;
       // If no instruction from the Ready List can be
       // scheduled in the current cycle, advance the clock
       // to the next cycle, and recompute the Ready List
       Schedule that instruction;
       Add any new instructions that have now become ready to
           the Ready List;
       For Each Instruction I in the Ready List,
           Compute Slack(I);
       Sort the Ready List according to Slack;
}
```


RAS on SpecInt2000 at base



Summary

- New RAS algorithm
- Better schedules for resource-constrained scheduling regions
- Implemented in the post-pass scheduler of the Intel® Itanium® compiler
- Improved schedules seen, but no measurable gain in performance